Pico-8 API Cheatsheet Version 0.1.10

Introduction

I've been working with the wonderful Pico-8 fantasy console for almost a week now and have to look into the API from time to time. Neko250 has created a fancy website which is licensed under the Creative Commons Attribution 4.0 International License. Unfortunately, I can only use the website for the Pico-8 API with JavaScript, but I have disabled it because of my Alligator Browser. I hereby port the API to a single subpage in my web project so that it can be read in my own browser (which I want to use more in the future). This will look less pretty here, but it will work better in a text browser (e.g. Lynx or W3M).

Root

Pico-8 Specs

- - display: 128x128, fixed 16 colour palette
- - input: 6 buttons
- - cartridge size: 32k
- - sound: 4 channel, 64 definable chip blerps
- - code: lua, max 8192 tokens of code
- - sprites: single bank of 128 8x8 sprites + 128 shared
- - map: 128x32 8-bit cels + 128x32 shared

General

- fullscreen: `alt + enter` / `cmd + f`
- quit: `alt + f4` / `cmd + q`
- reload and run: `ctrl + r`
- save: `ctrl + s`
- mute / unmute: `ctrl + m`
- show fps: `ctrl + 1`
- screenshot: `ctrl + 6` / `f6` / `f1`
- cart img: `ctrl + 7` / `f7` / `f2`
- start video: `ctrl + 8` / `f8` / `f3`
- save video: `ctrl + 9` / `f9` / `f4`
- keyboard cursor: `f10`
- navigate editors: `alt + right` / `alt + left`
- cli completion: `tab`

Splore

- navigate lists: `left` / `right`
- navigate carts: `up` / `down` / `home` / `end` / `pageup` / `pagedown`
- launch cart: `x` / `o` / `menu`
- favourite cart: `f`

Sprite / Map Editor

- undo: `ctrl + z`
- copy (bbs support): `ctrl + c`
- paste (bbs support): `ctrl + v`
- pan: `space` / `mmb`
- navigate sprites: `q` / `w`
- navigate colours: `1` / `2`
- navigate tabs: `+` / `-`
- fullscreen: `tab`
- zoom: `mousewheel`
- flip y<: `f`
- flip x: `v`
- rotate: `r`
- move: `cursors`

draw tool

- replace colour: `lctrl`
- eyedropper: `rmb`

stamp tool

- stamp w/ transparency: `lctrl`

select tool

- select sprites: `shift + drag`

Command Line Interface

- assert(condition) `-- verify that condition is true`
- cd .. `-- change to parent directory`
- cd [dirname] `-- change directory`
- dir() `-- list items`
- exit() `-- close pico-8 app`
- export(filename.html, [cart1.p8, cart2.p8 ... cart15]) `-- export game in html; up to 15 extra carts (.p8 format)`
- export(filename.png) `-- export spritesheet`
- export(filename.wav) `-- export the current sfx / music`
- export(filename%d.wav) `-- export all sfx as numbered files`
- extcmd(cmd) `-- control screenshots; "label" = cart label; "screen" = screenshot; "rec" = start gif; "video" = save gif`
- folder() `-- open carts folder in operating system`
- help() `-- show summary of system commands`
- import(filename.png) `-- import spritesheet. expects 128x128 png and colour-fits to the pico-8 palette`
- info() `-- print cart info`
- install_demos() `-- install demo p8 carts alongside pico-8 app`
- install_games() `-- install selection of bbs carts`
- keyconfig() `-- keyboard configuration for player buttons`
- load("@clip") `-- paste cart from clipboard, bbs support`
- load(filename) `-- load cart; works with multi-cart export`
- ls() `-- list items`
- menuitem(index, [label, callback]) `-- add an extra item to the pause menu; index in [1..5]; no label or callback removes the item`
- mkdir(dirname) `-- create directory`
- printh(str, [filename, overwrite]) `-- print str to terminal; append or overwrite to filename`
- reboot() `-- reboot pico8`
- resume() `-- resume cart execution`
- run() `-- boot cart`
- save("@clip") `-- copy cart to clipboard, bbs support`
- save(filename) `-- save cart`
- shutdown() `-- close pico-8 app`
- splore() `-- explore cartridges`
- stat(x) `-- read some execution values, read below`
- time() `-- returns seconds since last reboot`
- type(v) `-- returns type of v: number, string or table`

Code Editor

- skip across words: `ctrl + left` / `ctrl + right`
- select: `shift`
- select word: `double click`
- cut: `ctrl + x`
- copy: `ctrl + c`
- paste: `ctrl + v`
- undo: `ctrl + z`
- redo: `ctrl + y`
- indent: `tab`
- unindent: `shift + tab`
- duplicate line: `ctrl + d`
- search: `ctrl + f`
- repeat search: `ctrl + g`
- navigate functions: `alt + up` / `alt + down`
- start / end of code: `ctrl + up` / `ctrl + down`
- button glyphs: `shift + l, r, u, d, x, o`
- icons: `shift + qwetyipasfghjkzcvbnm`
- glyph mode: `ctrl + k`

SFX / Music Editor

- play / pause: `space`
- copy: `ctrl + c`
- paste: `ctrl + v`
- set all notes: `shift + lmb`
- modify speed: `<` / `>`
- navigate: `home`, `end`, `pageup`, `pagedown`, `mousewheel`
- navigate patterns: `+` / `-`
- snap to Cm pentatonic: `ctrl`

- speed x4: `shift`
- release loop: `a`
- delete: `backspace`

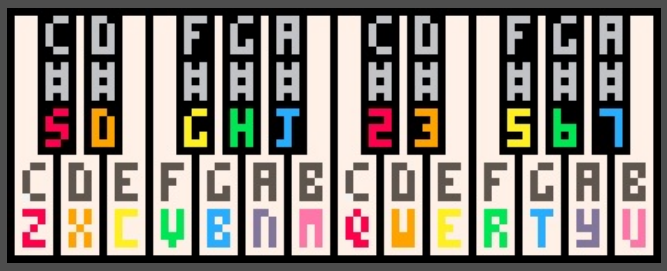
effects

- 0 :: none
- 1 :: slide
- 2 :: vibrato
- 3 :: drop (drum kick!)
- 4 :: fade in
- 5 :: fade out
- 6 :: fast arpeggio (4 notes)
- 7 :: slow arpeggio (4 notes)

Pico-8 Manpage

- pico-8 [switches] [filename.p8] `# run pico-8 from the system cli`
- -run `# boot filename.p8 on startup`
- -width n `# set the window or screen width and adjust scale to fit if not specified`
- -height n `# set the window or screen height and adjust scale to fit if not specified`
- -windowed b `# set windowed mode off (0) or on (1)`
- -sound n `# sound volume [0..256]`
- -music n `# sound volume [0..256]`
- -joystick n `# joystick controls starts at player n [0..7]`
- -pixel_perfect b `# set filter for screen stretching off (0) or on (1)`
- -draw_rect x,y,w,h `# window coordinates and size`
- -splore `# boot in splore mode`
- -home path `# customise home directory`
- -desktop path `# set desktop path for screenshots and gifs`
- -screenshot_scale n `# size of each screenshot pixel; default = 3`
- -gif_scale n `# size of each gif pixel; default = 2`
- -gif_len n `# set the maximum gif length in seconds; n in [1..120]`
- -gui_theme n `# set the code editor theme; 0 = classic, 1 = dark blue`
- -timeout n `# timeout switch for splore downloads`
- -frameless b `# set borderless window off (0) or on (1)`
- -show_fps b `# set fps display off (0) or on (1)`

Audio



- music([n, [fade_len, [channel_mask]]]) `-- play music; n = -1: stop`
- sfx(n, [channel, [offset]]) `-- play sfx; n = -1: stop in channel; n = -2: release loop in channel`

Cart Data

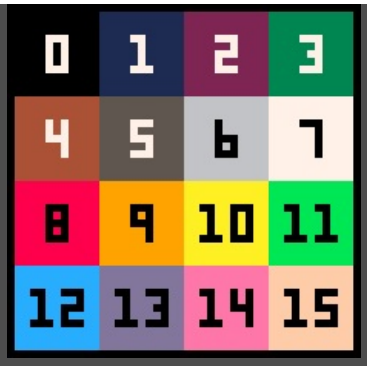
- cartdata(id) `-- open cart data named id (once per execution!)`
- dget(idx) `-- get number at index`
- dset(idx, val) `-- set number at index to value`

Coroutines

Coroutines

- cocreate(f) `-- returns a coroutine that executes f`
- coresume(c) `-- resume coroutine c execution`
- costatus(c) `-- returns true if c is still executing, false otherwise`
- yield() `-- use inside a coroutine; makes a coroutine pause execution until next resume`

Graphics



No.	Hex-Code	RGB	Name
0	#000000	(0, 0, 0)	Black
1	#000000	(29, 43, 83)	Dark-Blue
2	#7E2553	(126, 37, 83)	Dark-Purple
3	#008751	(0, 135, 81)	Dark-Green
4	#AB5236	(171, 82, 54)	Brown
5	#5F574F	(95, 87, 79)	Dark-Gray
6	#C2C3C7	(194, 195, 199)	Light-Gray
7	#FFF1E8	(255, 241, 232)	White
8	#FF004D	(255, 0, 77)	Red
9	#FFA300	(255, 163, 0)	Orange
10	#FFEC27	(255, 236, 39)	Yellow
11	#00E436	(0, 228, 54)	Green
12	#29ADFF	(41, 173, 255)	Blue
13	#83769C	(131, 118, 156)	Indigo
14	#FF77A8	(255, 119, 168)	Pink
15	#FFCCAA	(255, 204, 170)	Peach

Source: Roman Zolotarev's PICO-8 palette reference

- camera([x, y]) `-- set camera position`
- circ(x, y, r, [col]) `-- draw circle`
- circfill(x, y, r, [col]) `-- draw filled circle`
- clip([x, y, w, h]) `-- set screen clipping region`
- cls([col]) `-- clear screen; col = clear color`
- color(col) `-- set default color`
- cursor(x, y) `-- set cursor and CR/LF margin position`
- fget(n, [f]) `-- get values of sprite flags`
- flip() `-- flip screen back buffer (30fps)`
- fset(n, [f], v) `-- set values of sprite flags`
- line(x0, y0, x1, y1, [col]) `-- draw line`
- pal(c0, c1, [p]) `-- switch colour 0 to colour 1; p = 0 = draw palette; p = 1 = screen palette`
- palt(col, t) `-- set transparency for colour to t (bool)`
- pget(x, y) `-- get pixel colour`
- print(str, [x, y, [col]]) `-- print string`
- pset(x, y, [col]) `-- set pixel colour`
- rect(x0, y0, x1, y1, [col]) `-- draw rectangle`
- rectfill(x0, y0, x1, y1, [col]) `-- draw filled rectangle`
- sget(x, y) `-- get spritesheet pixel colour`
- spr(n, x, y, [w, h], [flip_x], [flip_y]) `-- draw sprite`
- sset(x, y, [col]) `-- set spritesheet pixel colour`
- sspr(sx, sy, sw, sh, dx, dy, [dw, dh], [flip_x], [flip_y]) `-- draw texture from spritesheet`
- fillp(mask) `-- set fill pattern for circ, circfill, rect, rectfill, pset, and line`

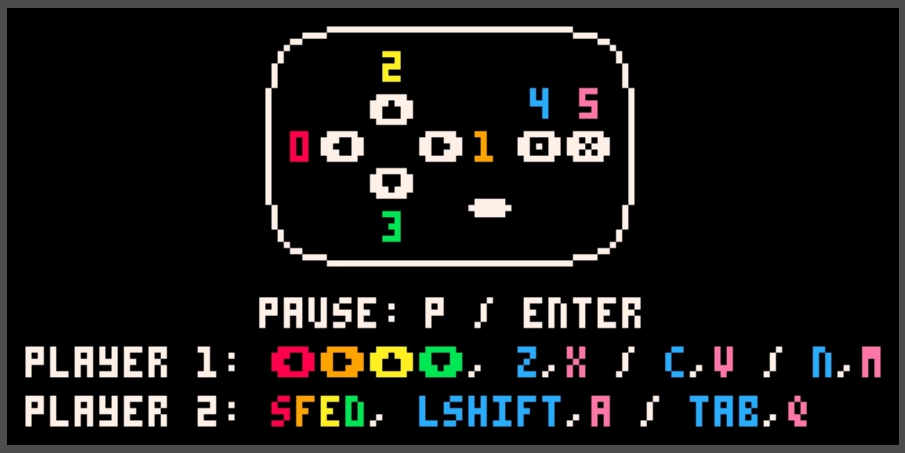
fillp() Mask

```
.-----------------------.
|32768|16384| 8192| 4096|
|-----|-----|-----|-----|
| 2048| 1024| 512 | 256 |
|-----|-----|-----|-----|
| 128 |  64 |  32 |  16 |
```

```
|-----|-----|-----|-----|
|  8  |  4  |  2  |  1  |
'-----------------------'
```

Input

Input Manipulation



- btn([i, [p]]) `-- get button i state for player p`
- btnp([i, [p]]) `-- only true when the button was not pressed the last frame; repeats every 4 frames after button held for 15 frames`

Map

Map Manipulation

- map(cel_x, cel_y, sx, sy, cel_w, cel_h, [layer] `-- draw map; layers from flags; sprite 0 is empty`
- mapdraw(cel_x, cel_y, sx, sy, cel_w, cel_h, [layer]) `-- same as 'map()'; draw map`
- mget(x, y) `-- get map value`
- mset(x, y, v) `-- set map value`

Math

Math Functions

- -32768.0 .. 32767.99 `-- numeric representation range`
- abs(x) `-- x absolute value`
- atan2(dx, dy) `-- convert (dx, dy) to an angle in [0..1]`
- band(x, y) `-- bitwise conjunction`
- bnot(x) `-- bitwise negation`
- bor(x, y) `-- bitwise disjunction`
- bxor(x, y) `-- bitwise exclusive disjunction`
- cos(x) `-- x cosine, [0..1]`
- flr(x) `-- round down`
- -flr(-x) `-- not a function per se, but will work as ceil(x)`
- max(x, y) `-- x/y maximum`
- mid(x, y, z) `-- x/y/z middle value`
- min(x, y) `-- x/y minimum`
- rnd(x) `-- random; 0 <= n < x`
- sgn(x) `-- returns argument sign: -1 or 1; sgn(0) = 1`
- shl(x, y) `-- shift left`
- shr(x, y) `-- shift right`
- sin(x) `-- x sine, [0..1]; inverted`
- sqrt(x) `-- x square root`
- srand(x) `-- set random seed`

Operators

- a + b `-- sum`
- a - b `-- sub`
- a * b `-- mul`
- a / b `-- div`
- a % b `-- mod`
- a ^ b `-- pow`
- a += b `-- sum to`
- a -= b `-- sub to`
- a *= b `-- mul to`

- a /= b `-- div to`
- a %= b `-- mod to`
- a ^= b `-- pow to`
- a == b `-- compare equals`
- a ~= b `-- compare not equals`
- a != b `-- compare not equals`
- a > b `-- compare greater than`
- a < b `-- compare less than`
- a >= b `-- compare greater than or equals`
- a <= b `-- compare less than or equals`
- not a `-- boolean negation`
- a and b `-- boolean conjunction`
- a or b `-- boolean disjunction`
- (a) `-- operation delimiters`

Flow Control

```
::label:: -- label
goto label -- jump

if (<condition>) -- inline if

if <condition> then
 -- if block
elseif <condition> then
 -- elseif block
else
 -- else block
end

while <condition> do
 -- while block
end

repeat
 -- repeat block
until <condition>

for <var> = <first>, <last>, <step> do
 -- for block
end

for <var> in all(<table>) do
 -- for block
end

for <key>, <value> in pairs(<table>) do
 -- for block
end
```

Memory

Memory Manipulation

- cstore(dest_addr, src_addr, len, [filename]) `-- copy bytes from ram to rom [load from filename; works with multi-cart export]`
- memcpy(dest_addr, src_addr, len) `-- copy bytes`
- memset(dest_addr, val, len) `-- set len bytes to val`
- peek(addr) `-- read byte in ram address`
- poke(addr, val) `-- write val in ram address`
- reload(dest_addr, src_addr, len, [filename]) `-- copy bytes from rom to ram [load from filename; works with multi-cart export]`

Memory Types

1. base ram (32kB)
2. cart rom
3. lua ram (1MB)

RAM Layout

- 0x1000 - gfx2/map2 (shared)
- 0x2000 - map
- 0x3000 - gfx_props
- 0x3100 - song
- 0x3200 - sfx
- 0x4300 - user data
- 0x5e00 - persistent cart data (256 bytes)
- 0x5f00 - draw state

- 0x5f40 - hardware state
- 0x5f80 - gpio pins (128 bytes)
- 0x6000 - screen (8k)

Peek/Poke

Devkit Input Mode

- poke(0x5f2d, 1) `-- enable devkit input mode`
- stat(30) `-- read keyboard had input (bool); repeats every 4 frames after key held for 15 frames`
- stat(31) `-- read keyboard character`
- stat(32) `-- read x coord`
- stat(33) `-- read y coord`
- stat(34) `-- read button bitmask; 1 = primary, 2 = secondary, 4 = middle`

Extra Graphics Mode

- poke(0x5f2c, 0) `-- standard, 128x128`
- poke(0x5f2c, 1) `-- horizontal stretch, 64x128`
- poke(0x5f2c, 2) `-- vertical stretch, 128x64`
- poke(0x5f2c, 3) `-- zoomed, 64x64`
- poke(0x5f2c, 4) `-- standard, 128x128`
- poke(0x5f2c, 5) `-- mirror left half`
- poke(0x5f2c, 6) `-- mirror top half`
- poke(0x5f2c, 7) `-- mirror top-left quarter`

Raspberry Pi / Pocketchip GPIO

- `-- run pico-8 as root ("sudo pico-8")`
- `-- 128 pin values in the range [0..255]`
- poke(0x5f80, value) `-- send value to gpio0`
- peek(0x5f81) `-- get value from gpio1`
- `-- get value from gpio1`

Pico-8

Execution Flow

- _draw() `-- called once per visible frame`
- _init() `-- called once on program startup`
- _update() `-- called once per update at 30fps`
- _update60() `-- called once per update at 60fps`

Stats

- stat(0) `-- memory usage in [0..1024]`
- stat(1) `-- cpu usage; 1.0 == 100% cpu at 30fps`
- stat(4) `-- clipboard; after user pressed ctrl-v`
- stat([16..19]) `-- index of playing sfx on channels [0..3]`
- stat([20..23]) `-- note number (0..31) on channels [0..3]`
- stat(30) `-- keyboard key hit; see "peek / poke" tab`
- stat(31) `-- keyboard character; see "peek / poke" tab`
- stat(32) `-- mouse x coord; see "peek / poke" tab`
- stat(33) `-- mouse y coord; see "peek / poke" tab`
- stat(34) `-- mouse button bitmask; see "peek / poke" tab`

Javascript

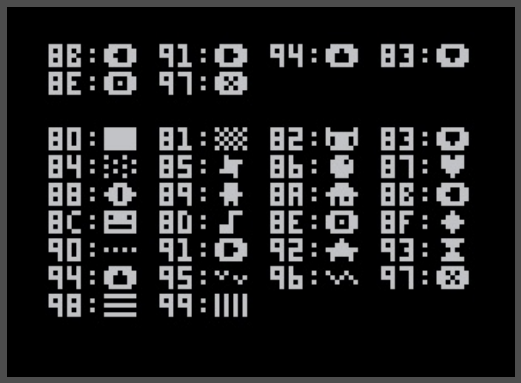
- pico8_gpio[] `// read and write gpio pins`
- pico8_buttons[] `// bitfields for player input`

Private Functions (use carefully)

- _get_frames_skipped() `-- used automatically by _mainloop()`
- _get_menu_item_selected(n) `-- returns true or false if the n-th menu item is selected; n in [1..5]; used automatically by _mainloop()`
- _mainloop() `-- main pico-8's "while true"`
- _pausemenu[n].callback() `-- execute the n-th menu item's callback; n in [1..5]; used automatically by _mainloop()`
- _set_mainloop_exists(n) `-- ?`
- _update_buttons() `-- used automatically by _mainloop()`
- holdframe() `-- used automatically by _mainloop()`

Strings

String Manipulation



- #s -- string length
- "three " .. 4 `-- string concatenation`
- sub(str, from, [to]) `-- substring`
- tostr(val, [hex]) `-- convert val to a string; if hex is true and val is a number, output format is "0x0000.0000"`
- tonum(str) `-- cast parseable str to number; nil if str isn't a number; "0xAF" format for hex`
- "123.45" + 0 `-- alternative to tonum`

special characters

```
print("l:\x8b r:\x91 u:\x94 d:\x83 o:\x8e x:\x97")
```

Tables

Table Manipulation

- add(t, v) `-- add v to t`
- all(t) `-- used in 'for v in all(t)' loops`
- count(t) `-- returns number of elements in the table`
- del(t, v) `-- delete first instance of v in t`
- foreach(t, f) `-- call f() for each v in t`
- pairs(t) `-- used in 'for k,v in pairs(t)' loops`

Metatables

- mt = {} `-- init metatable`
- setmetatable(t, mt) `-- set metatable mt to table t`

Metatables :: Special Keys

```
-- object oriented programming; invoked as 't:foo()' / 't:bar()'
fun = {}
function fun.foo(t) dosomething() end
function fun.bar(t) dosomethingmore() end
mt.__index = fun

-- property assignment; invoked when calling 't[k] = v'
function mt.__newindex(t, k, v) rawset(t, k, v) end

-- weak references; keys and / or values
mt.__mode = "k" or "v" or "kv"

-- treat a table like a function; invoked when calling 't()'
function mt.__call(t) dosomething() end

-- hide metatable; returned by 'getmetatable(t)'
public_mt = {}
mt.__metatable = public_mt

-- string casting; returned by 'tostring(t)'
function mt.__tostring(t) return tostring(t) end

-- table length; returned by '#t'
function mt.__len(t) return #t end
```

Metatables :: Mathematic Operators

```
-- unary minus; returned by '-t'
function mt.__unm(t) return -t end

-- addition; returned by 'a + b'
function mt.__add(a, b) return a + b end
```

```
-- subtraction; returned by 'a - b'
function mt.__sub(a, b) return a - b end

-- multiplication; returned by 'a * b'
function mt.__mul(a, b) return a * b end

-- division; returned by 'a / b'
function mt.__div(a, b) return a / b end

-- modulo; returned by 'a % b'
function mt.__mod(a, b) return a % b end

-- involution; returned by 'a ^ b'
function mt.__pow(a, b) return a ^ b end

-- concatenation; returned by 'a .. b'
function mt.__concat(a, b) return a .. b end
```

Metatables :: Comparison Operators

```
-- check for equality; returned by 'a == b'
function mt.__eq(a, b) return a == b end

-- check for less-than; returned by 'a < b'
function mt.__lt(a, b) return a < b end

-- check for less-than-or-equal; returned by 'a <= b'
function mt.__le(a, b) return a <= b end
```